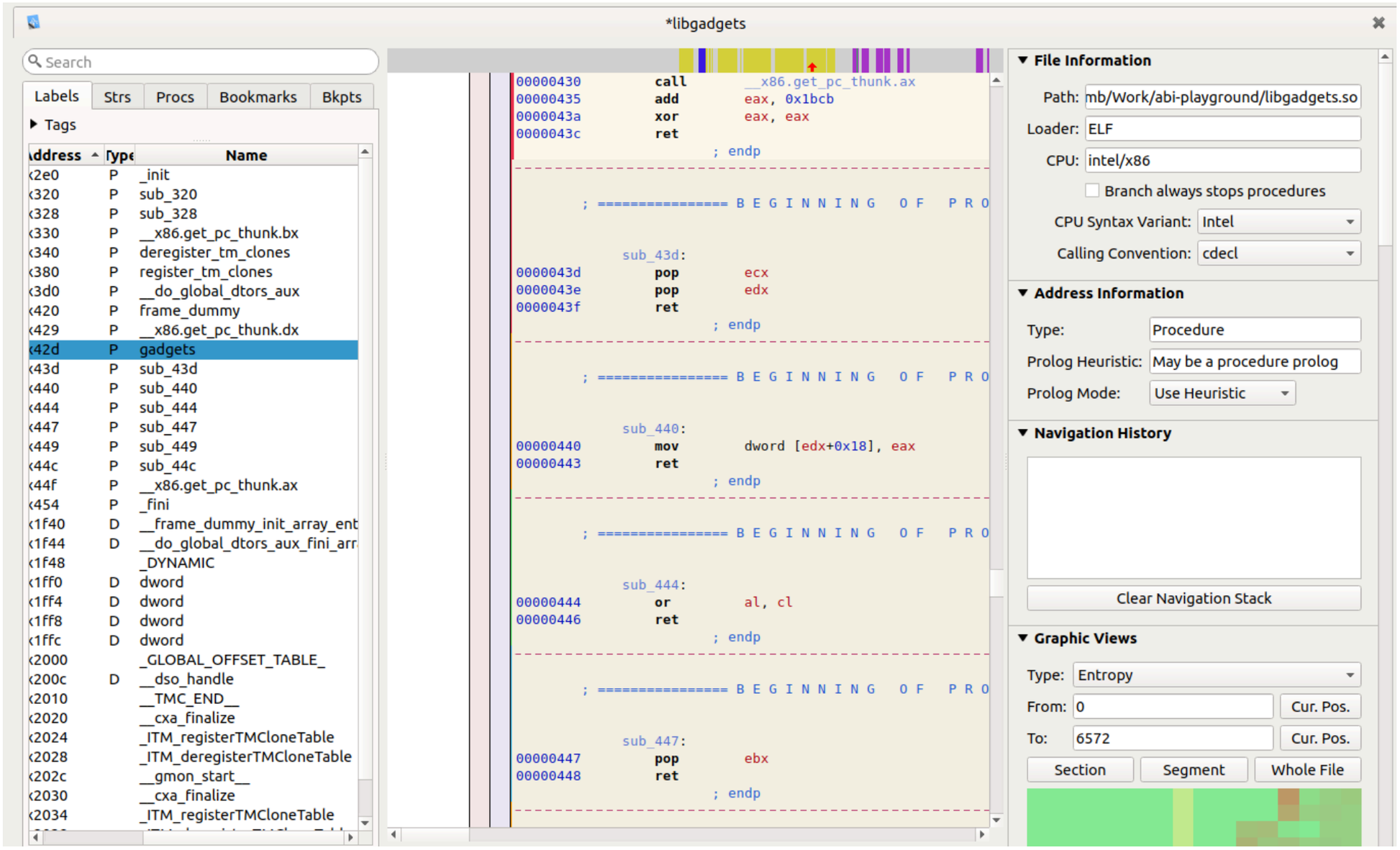
# Technical Cybersecurity

## Finding Gadgets




# Disassemble

Looks like our gadgets are there!

```
cclamb@ubuntu:~/Work $ git clone https://github.com/packz/ropeme.git
Cloning into 'ropeme'...
remote: Enumerating objects: 30, done.
remote: Total 30 (delta 0), reused 0 (delta 0), pack-reused 30
Unpacking objects: 100% (30/30), done.
cclamb@ubuntu:~/Work $ mkvirtualenv ropeme
New python executable in /home/cclamb/.virtualenvs/ropeme/bin/python
Installing setuptools, pip, wheel...
done.
virtualenvwrapper.user_scripts creating /home/cclamb/.virtualenvs/ropeme/bin/predeactivate
virtualenvwrapper.user_scripts creating /home/cclamb/.virtualenvs/ropeme/bin/postdeactivate
virtualenvwrapper.user_scripts creating /home/cclamb/.virtualenvs/ropeme/bin/preactivate
virtualenvwrapper.user_scripts creating /home/cclamb/.virtualenvs/ropeme/bin/postactivate
virtualenvwrapper.user_scripts creating /home/cclamb/.virtualenvs/ropeme/bin/get_env_details
(ropeme) cclamb@ubuntu:~/Work $ pip install diStorm3
Collecting diStorm3
Installing collected packages: diStorm3
Successfully installed diStorm3-3.3.8
(ropeme) cclamb@ubuntu:~/Work $ ropeme/ropeme/ropshell.py
Simple ROP interactive shell: [generate, load, search] gadgets
ROPeMe>
```

# ROPeMe

Long Le Dinh, Presented at Blackhat 2010

rop.c

```c
#include <string.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <unistd.h>
#include <sys/stat.h>

#define BUF_SIZE 5

void smash(char* arg) {
  char buffer[BUF_SIZE];
  strcpy(buffer, arg);
}

void load(void) {
  struct stat st;
  stat("./libgadgets.so", &st);
  int fd = open("./libgadgets.so", 0, 00);
  mmap((void*) 0x30000000, st.st_size, PROT_READ, 17, fd, 0);
}

int main(int argc, char* argv[]) {
  char* arg = argv[1];
  load();
  smash(arg);
  return 0;
}
```

# Remember cheating?

Statically loading the library into address 0x30000000

# FIND GADGETS

- We're looking for the gadgets we previously programmed into the lib gadget.so library
- And we find them
- …including some extras!

```
(ropeme) cclamb@ubuntu:~/Work/abi-playground $ ropshel
Simple ROP interactive shell: [generate, load, search]
ROPeMe> generate libgadgets.so
Generating gadgets for libgadgets.so with backward dep
It may take few minutes depends on the depth and file
Processing code block 1/1
Generated 83 gadgets
Dumping asm gadgets to file: libgadgets.so.ggt ...
OK
ROPeMe> load libgadgets.so.ggt
Loading asm gadgets from file: libgadgets.so.ggt ...
Loaded 83 gadgets
ELF base address: 0x0
OK
ROPeMe> search xor eax, eax
Searching for ROP gadget:  xor eax, eax with constrain
0x43aL: xor eax eax ;;

ROPeMe> search pop ecx % pop edx
Searching for ROP gadget:  pop ecx % pop edx with cons
0x43dL: pop ecx ; pop edx ;;

ROPeMe> search mov [ edx + 0x18 ] eax
Searching for ROP gadget:  mov [ edx + 0x18 ] eax with
0x440L: mov [edx+0x18] eax ;;

ROPeMe> search or al, cl
Searching for ROP gadget:  or al, cl with constraints:
0x444L: or al cl ;;

ROPeMe> search pop ebx
Searching for ROP gadget:  pop ebx with constraints: [
0x301L: pop ebx ;;
0x447L: pop ebx ;;
0x466L: pop ebx ;;

ROPeMe> search int 0x80 %
Searching for ROP gadget:  int 0x80 % with constraints
0x449L: int 0x80 ;;

ROPeMe>
```

# Gadgets?

## Find in ROP

- Ropeme gives you an offset; add the offset to the location of lib gadget.so (0x30000000)
- Check to see if the addresses are really there

```
Dump of assembler code for function main:
   0x08048545 <+0>:     lea    ecx,[esp+0x4]
   0x08048549 <+4>:     and    esp,0xfffffff0
   0x0804854c <+7>:     push   DWORD PTR [ecx-0x4]
   0x0804854f <+10>:    push   ebp
   0x08048550 <+11>:    mov    ebp,esp
   0x08048552 <+13>:    push   ecx
   0x08048553 <+14>:    sub    esp,0x14
   0x08048556 <+17>:    call   0x804858b <__x86.get_pc_th
   0x0804855b <+22>:    add    eax,0x1aa5
   0x08048560 <+27>:    mov    eax,ecx
   0x08048562 <+29>:    mov    eax,DWORD PTR [eax+0x4]
   0x08048565 <+32>:    mov    eax,DWORD PTR [eax+0x4]
   0x08048568 <+35>:    mov    DWORD PTR [ebp-0xc],eax
   0x0804856b <+38>:    call   0x80484e1 <load>
   0x08048570 <+43>:    sub    esp,0xc
   0x08048573 <+46>:    push   DWORD PTR [ebp-0xc]
   0x08048576 <+49>:    call   0x80484b6 <smash>
   0x0804857b <+54>:    add    esp,0x10
   0x0804857e <+57>:    mov    eax,0x0
   0x08048583 <+62>:    mov    ecx,DWORD PTR [ebp-0x4]
   0x08048586 <+65>:    leave
   0x08048587 <+66>:    lea    esp,[ecx-0x4]
   0x0804858a <+69>:    ret
End of assembler dump.
(gdb) b *0x08048570
Breakpoint 1 at 0x8048570: file rop.c, line 24.
(gdb) r
Starting program: /home/cclamb/Work/abi-playground/rop

Breakpoint 1, main (argc=1, argv=0xffffcea4) at rop.c:24
24          smash(arg);
(gdb) x/i 0x3000043a
   0x3000043a:  xor    eax,eax
(gdb) x/i 0x3000043d
   0x3000043d:  pop    ecx
(gdb) x/2i 0x3000043d
   0x3000043d:  pop    ecx
   0x3000043e:  pop    edx
(gdb) x/3i 0x3000043d
   0x3000043d:  pop    ecx
   0x3000043e:  pop    edx
   0x3000043f:  ret
(gdb)
```

Stack formatting next.